IDE command-line switches

Turbo C++'s integrated development environment (IDE) offers several command-line switches, which use this syntax:

TC [*sourcename*|*projectname*] [*option* [*option* . . .]]

where *sourcename* is any ASCII file, *projectname* is your project file (it *must* have the .PRJ extension), and *option* can be any one of the switches.

Switch	Function
/b	Recompiles and links all files in your project, prints compiler messages to standard output device, and returns to DOS. Can invoke TC from a batch file. Use **/b** as `tc /b` or `tc myproj.prj /b`, where myproj.prj is your project file name.
/m	Does a make rather than a build. Can use like **/b**.
/d	Uses dual monitor mode if appropriate hardware detected. Use dual monitor mode when you run or debug a program or shell to DOS (**F**ile|**D**OS Shell).
/e, **/x**	Normally, Turbo C++ swaps to a hard disk when allocating memory. If you have expanded or extended memory, use the **/e** or **/x** switches respectively to improve performance.
/r*x*	Use this switch if all your extended or expanded memory has been allocated to a RAM disk. *x* is the letter of the "fast" swap drive.
/l	Use this switch if you're running Turbo C++ on an LCD screen.
/p	Use this switch, which controls palette swapping on EGA video adapters, when your program modifies the EGA palette registers. The EGA palette will be restored each time the screen is swapped.

MAKE command-line options

The general syntax for MAKE is

make [*option* [*option* . . .]] [*target* [*target* . . .]]

Option	What it does
-? or **-h**	Prints a help message. The default options are displayed with plus signs following.
-a	Causes an automatic dependency check on .OBJ files.
-B	Builds all targets regardless of file dates.
-D*identifier*	Defines the named identifier to the string consisting of the single character 1 (one).
-D*iden*=*string*	Defines the named identifier *iden* to the string after the equal sign. The string cannot contain any spaces or tabs.
-f*filename*	Uses *filename* as the MAKE file. If *filename* does not exist and no extension is given, tries FILENAME.MAK.
-i	Does not check (ignores) the exit status of all programs run. Continues regardless of exit status. This is equivalent to putting '-' in front of all commands in the MAKEFILE.
-I*directory*	Searches for include files in the indicated directory (as well as in the current directory).
-K	Keeps (does not erase) temporary files created by MAKE. All temporary files have the form MAK*nnnn*.$$$, where *nnnn* ranges from 0000 to 9999.
-n	Prints the commands but does not actually perform them. This is useful for debugging a makefile.
-s	Does not print commands before executing. Normally, MAKE prints each command as it is about to be executed.
-S	Swaps MAKE out of memory while executing commands. This significantly reduces the memory overhead of MAKE, allowing it to compile very large modules.
-U*identifier*	Undefines any previous definitions of the named identifier.
-W	Writes the current specified non-string options (like -s and -a) to MAKE.EXE.

IDE hot keys

Many menu items have corresponding *hot keys*: one- or two-key shortcuts that immediately activate that command or dialog box. Except *for Shift-F1*, all shifted function keys are used for transfer hot keys.

Key(s)	Menu item	Function
F1	Help	Displays a help screen.
F2	File | Save	Saves the file that's in the active Edit window.
F3	File | Open	Brings up a dialog box so you can open a file.
F4	Run | Go to Cursor	Runs your program to the line where the cursor is positioned.
F5	Window | Zoom	Zooms the active window.
F6	Window | Next	Cycles through all open windows.
F7	Run | Trace Into	Runs your program in debug mode, tracing into functions.
F8	Run | Step Over	Runs your program in debug mode, stepping over function calls.
F9	Compile | Make EXE	Makes the current window or project.
F10	(none)	Takes you to the menu bar.

Color table

CGA		EGA/VGA	
Name	**Value**	**Name**	**Value**
BLACK	0	EGA_BLACK	0
BLUE	1	EGA_BLUE	1
GREEN	2	EGA_GREEN	2
CYAN	3	EGA_CYAN	3
RED	4	EGA_RED	4
MAGENTA	5	EGA_MAGENTA	5
BROWN	6	EGA_LIGHTGRAY	7
LIGHTGRAY	7	EGA_BROWN	0
DARKGRAY	8	EGA_DARKGRAY	56
LIGHTBLUE	9	EGA_LIGHTBLUE	57
LIGHTGREEN	10	EGA_LIGHTGREEN	58
LIGHTCYAN	11	EGA_LIGHTCYAN	59
LIGHTRED	12	EGA_LIGHTRED	60
LIGHTMAGENTA	13	EGA_LIGHTMAGENTA	61
YELLOW	14	EGA_YELLOW	62
WHITE	15	EGA_WHITE	63

Note that valid colors depend on the current graphics driver and current graphics mode. Changes made to the palette are seen immediately onscreen. Each time a palette color is changed, all occurrences of that color onscreen will change to the new color value.

1800 GREEN HILLS ROAD, P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001
UNIT 8 PAVILION, RUSCOMBE BUSINESS PARK, TWYFORD, BERKSHIRE, RG10 9NN-ENGLAND
43 AVENUE DE L'EUROPE—B.P. n6, 78143 VELIZY VILLACOUBLAY CEDEX, FRANCE
GYDEVANG 12, DK 3450—ALLEROD, DENMARK
BOX 1236, S-172 24 SUNDBYBERG, SWEDEN
PART # 14CL-CPP01 BOR 1768

TURBO C++
QUICK REFERENCE BOOKLET

Command-line compiler options

Invoke Turbo C++ from the command line by typing `TCC` at the DOS prompt, followed by command-line arguments. Here's the generic format:

tcc[*option* [*option* . . .]] *filename* [*filename* . . .]

Each command-line option is preceded by a hyphen (–) and separated from the TCC command, other options, and file names by one space. You can also use a configuration file.

Option	Function
@ *filename*	Response files
+ *filename*	Tell TCC to use the alternate configuration file *filename*
-A	Use only ANSI keywords
-A- or -AT	Use Turbo C++ keywords (default)
-AK	Use only Kernighan and Ritchie keywords
-AU	Use only UNIX keywords
-a	Align word
-a-	Align byte (default)
-B	Compile and call the assembler to process inline assembly code
-b	Make enums word-sized (default)
-C	Nested comments on
-c	Compile to .OBJ but do not link
-D*name*	Define *name* to the string consisting of the null character
-D*name*=*string*	Defines *name* to *string*
-d	Merge duplicate strings on
-d-	Merge duplicate strings off (default)
-E*filename*	Use *filename* as the assembler to use
-e*filename*	Link to produce *filename*.EXE
-f	Emulate floating point (default)
-f-	Don't do floating point
-ff	Fast floating point (default)
-ff-	Strict ANSI floating point
-f87	Use 8087 hardware instructions
-f287	Use 80287 hardware instructions
-G	Optimize for speed
-G-	Optimize for size
-g*n*	Warnings: stop after *n* messages
-I*pathname*	Directories for include files
-i*n*	Identifier length is *n*
-j*n*	Errors: stop after *n* messages
-K	Default character type unsigned
-K-	Default character type signed (default)
-k	Standard stack frame on (default)
-L*pathname*	Directories for libraries
-l*x*	Pass option *x* to the linker (can use more than one *x*)
-l-*x*	Suppress option *x* for the linker (can use more than one *x*)
-M	Instruct the linker to create a map file
-mc	Compile using compact memory model*
-mh	Compile using huge memory model*
-ml	Compile using large memory model*
-mm	Compile using medium memory model*
-mm!	Compile using medium model; assume DS != SS*
-ms	Compile using small memory model (default)*
-ms!	Compile using small model; assume DS != SS*
-mt	Compile using tiny memory model*
-mt!	Compile using tiny model; assume DS != SS*

Command-line compiler options, continued

Option	*Function*
-N	Check for stack overflow
-n*pathname*	Output directory
-O	Optimize jumps
-O-	No optimization
-o*filename*	Compile source file to *filename*.OBJ
-P	Perform a C++ compile regardless of source file extension
-p	Use Pascal calling convention
-p-	Use C calling convention (default)
-Qe	Instructs the compiler to use all available EMS memory (default)
-Qe-	Instructs the compiler to not use any EMS memory
-Qx	Instructs the compiler to use all available extended memory
-Qx=*nnnn*	Instructs the compiler to reserve *nnnn* Kbytes of extended memory for other programs, and to use the rest itself
-Qx-	Instructs the compiler to not use any extended memory (default)
-r	Use register variables on (default)
-r-	Suppresses the use of register variables.
-rd	Allow only declared register variables to be kept in registers
-S	Produce .ASM output file
-T*string*	Pass *string* as an option to TASM or assembler specified with -E
-T-	Remove all previous assembler options
-U*name*	Undefine any previous definitions of *name*
-u	Generate underbars on (default)
-V, -V*n*	C++ virtual tables
-v	Source debugging on
-vi	Controls expansion of inline functions
-w	Display warnings on
-w-	Display warnings off
-w*xxx*	Enable *xxx* warning message
-w-*xxx*	Disable *xxx* warning message
-X	Disable compiler autodependency output
-Y	Enable overlay code generation
-Yo	Overlay the compiled files
-y	Line numbers on
-Z	Enable register usage optimization
-zA*name*	Code class
-zB*name*	BSS class
-zC*name*	Code segment
-zD*name*	BSS segment
-zE*name*	Far segment
-zF*name*	Far class
-zG*name*	BSS group
-zH*name*	Far group
-zP*name*	Code group
-zR*name*	Data segment
-zS*name*	Data group
-zT*name*	Data class
-z*X*	Use default name for *X*
-1	Generate 80186 instructions
-1-	Generate 8088/8086 instructions (default)
-2	Generate 80286 protected-mode compatible instructions

For details about memory models, see Chapter 4 "Memory models, floating point, and overlays" in the *Programmer's Guide*.

scanf and printf format specifiers

Here is the syntax for both **printf** and **scanf**:

printf %[*flags*][*width*][*precision*][*mod*]*type*
scanf %[*flags*][*width*][*mod*]*type*

Field	*printf*	*scanf*	*Description*
flags	-		Left-justify result
	+		Always prefix with + or -
	space		Prefix with a blank if non-negative
	#		Alternate form conversion
		*	Suppresses assignment of next field
width	**n**		Prints at least *n* characters, pad with blanks
	On		Prints at least **n** characters, pad with zeroes
	*		Next argument specifies width
		n	Maximum number of characters that will be read
precision	(default)		= 1 for **d,i,o,u,x,X** types = 6 for **e,E,f** types
	.0		No decimal point for **e,E,f** types
	.n		*n* decimal places or characters are printed
	*		Next argument specifies width
mod	**F**	**F**	Argument is a far pointer
	N	**N**	Argument is a near pointer
	h	**h**	**short int** for **d,i,o,u,x,X** types
	l	**l**	**long int** for **d,i,o,u,x,X** types **double** for **e,E,f,g,G** types
	L	**L**	**long int** for **d,i,o,u,x,X** types **long double** for **e,E,f,g,G** types
type	**c**	**c**	Single character
	d	**d**	**signed** decimal **int**
		D	**signed long** decimal **int**
	e,E	**e,E**	**signed** exponential
	f	**f**	**signed** floating point
	g,G	**g,G**	same as **e** or **f** based on value and precision
	i		**signed** decimal **int**
		i	**signed** decimal, octal, or hex **int**
		l	**signed** decimal, octal, or hex **long int**
	n	**n**	pointer to **int**
	o	**o**	**unsigned** octal **int**
		O	**unsigned** octal **long int**
	p	**p**	pointer
	s	**s**	string pointer
	u	**u**	**unsigned** decimal **int**
		U	**unsigned** decimal **long int**
	x,X	**x**	**unsigned** hex **int**
		X	**unsigned** hex **long int**

Debugger format specifiers

Character	*Function*
C	**Character.** Shows special display characters for control characters (ASCII 0 through 31); by default, such characters are shown using the appropriate C escape sequences (\n, \t, and so on). Affects characters and strings.
S	**String.** Shows control characters (ASCII 0 through 31) as ASCII values using the appropriate C escape sequences. Since this is the default character and string display format, the **S** specifier is only useful in conjunction with the **M** specifier.
D	**Decimal.** Shows all integer values in decimal. Affects simple integer expressions as well as arrays and structures containing integers.
H or X	**Hexadecimal.** Shows all integer values in hexadecimal with the 0*x* prefix. Affects simple integer expressions as well as arrays and structures containing integers.
F*n*	**Floating point.** Shows *n* significant digits (*n* is an integer between 2 and 18). The default value is 7. Affects only floating-point values.
M[*n*]	**Memory dump.** Displays a memory dump, starting with the address of the indicated expression. The expression must be a construct that would be valid on the left side of an assignment statement, i.e., a construct that denotes a memory address; otherwise, the **M** specifier is ignored. By default, each byte of the variable is shown as two hex digits. Adding a **D** specifier with the **M** causes the bytes to be displayed in decimal. Adding an **H** or **X** specifier causes the bytes to be displayed in hex. An **S** or a **C** specifier causes the variable to be displayed as a string (with or without special characters). The default number of bytes displayed corresponds to the size of the variable, but you can use a repeat count (*n*) to specify an exact number of bytes.
P	**Pointer**. Displays pointers in *seg:ofs* format with additional information about the address pointed to, rather than the default hardware-oriented *seg:ofs* format. Specifically, it tells you the region of memory in which the segment is located, and the name of the variable at the offset address, if appropriate. The memory regions are as follows:

Memory region	Evaluate message
0000:0000–0000:03FF	Interrupt vector table
0000:0400–0000:04FF	BIOS data area
0000:0500–Turbo C++	MS-DOS/TSR's
Turbo C++–User Program PSP	Turbo C++
User Program PSP	User Process PSP
User Program–top of RAM	Name of a static user variable if its address falls inside the variable's allocated memory; otherwise nothing
A000:0000–AFFF:FFFF	EGA/VGA Video RAM
B000:0000–B7FF:FFFF	Monochrome Display RAM
B800:0000–BFFF:FFFF	Color Display RAM
C000:0000–EFFF:FFFF	EMS Pages/Adaptor BIOS ROM's
F000:0000–FFFF:FFFF	BIOS ROM's

R	**Structure/Union**. Displays field names as well as values, such as {*X:1, Y:10, Z:5* }. Affects only structures and unions.